GWAPによるマイクロブログからの構造データの抽出

福角　駿† 森嶋　厚行††,††† 品川　徳秀††

† 筑波大学 情報学群情報メディア創成学類　〒305–8550 茨城県つくば市春日 1–2
†† 筑波大学 図書館情報メディア系/知的コミュニティ基盤研究センター　〒305–8550 茨城県つくば市春日 1–2
††† JST さきがけ
E-mail: †s0811544@u.tsukuba.ac.jp, ††{mori,siena}@slis.tsukuba.ac.jp

あらまし　近年，Twitter を代表としたマイクロブログを通じて発信される情報の量が増大している．しかし，これらは非構造データであるため，構造データと比較すると，その後の再利用や管理は難しい．一方，近年のネットワーク技術の発達により，クラウドソーシングなどを通じて人の力を利用したデータ処理が容易になりつつある．そこで，本稿ではマイクロブログを対象とし，GWAP (Game With a Purpose) を用いて非構造データから構造データを抽出する手法を提案する．GWAP とは，ゲームを行う副作用として，何らかの目的を達しようとするものである．本稿で提案する GWAP は，データ抽出に関する同調ゲームの副作用としてデータ抽出を行う．本ゲームの新規性は，一般的な同調ゲーム GWAP と異なり，プレイヤの入力として，データそのものだけでなく計算機がデータを抽出するためのルールも許可するような，新たな構造を持つ GWAP を提案していることである．

キーワード　マイクロブログ, クラウドソーシング

1. はじめに

近年，Twitter を代表としたマイクロブログを通じて発信される情報の量が増大している．東日本大震災においては，つくば市においても電車の営業停止，長期の断水被害などの影響があったが，電車の運行情報や給水所などについての情報源として役に立ったのは，マイクロブログなどの情報をまとめた「まとめサイト」であった．このマイクロブログに代表されるように，Web 上には役に立つ情報が多く存在するが，これらは非構造データであり，その後の再利用や管理が困難である．したがって，非構造データから構造データを抽出する数多くの研究がこれまでに行われてきている [1] [2].

非構造データから構造データを抽出する方法はいくつかに大別される．利用状況や対象の多彩さなどにより，それぞれ一長一短である．第一に，ラッパーなどの自動変換ソフトウェアを構築して抽出する方法がある．第二に，自然言語処理技術により構造データを抽出する方法がある．第三に，人手により構造データを抽出する方法がある．第三の方法には，まとめサイトの管理者のように特定の人物が行うものと Human Computation [3] のように複数の人物が参加して行うものがある．後者には，クラウドソーシングに代表されるように不特定多数の群衆が行うものから，比較的少数の協力者によって行われる手法まで含まれている．特にクラウドソーシングは近年注目を集めているアプローチである．

本研究では，第三のアプローチで利用可能な手法として，GWAP (Game With a Purpose) [4] により，Web 上の非構造データから構造データを抽出する手法を提案する．GWAP とは，ゲームを行う副作用として，何らかの目的を達しようとするものである．たとえば ESP Game [5] では，画像に対するキーワード当てゲームを用いて，画像のメタデータ収集を行う．

本研究では，データ抽出対象として，マイクロブログを主な対象とする．その理由は，ハッシュタグなどを利用して比較的同一テーマの情報を集めやすく，また，広範に多様なデータを収集するのが容易なため，構造データ抽出の適用例として向いていると考えられるからである．

GWAP によるデータ生成では，これまで，入力されたデータに関する同調ゲームが利用されることが一般的である．同調ゲームとは，ゲームに参加するプレイヤの振る舞いが同じであるときにポイントを与えるという構造を持つゲームである．本研究の新規性は，データそのものを入力するだけでなく，計算機がデータ抽出するためのルールを入力するという選択肢も用意した GWAP の可能性を提示することである．

本研究の貢献は次の通りである．第一に，非構造データから構造データの抽出問題に対して GWAP の適用が可能であることを示すことである．第二に，プレイヤが単にデータを入力するだけでなく，データ抽出のためのルールを入力するという選択肢を持つようなゲームを提案していることである．第三に，このゲームが合理的なプレイヤの元で正しく動作し，必ず終了することを示すことである．第四に，実験により，プレイヤがデータ抽出ルールという “メタなデータ” を入力することが，GWAP による構造データ抽出に対してどのような影響を及ぼすかを明らかにすることである．

本稿は次のように構成される．2 節では，関連研究・システムについて述べる．3 節では，提案システム Tweet Pecker について議論するために必要な準備を行う．4 節では，GWAP による構造データ抽出システム Tweet Pecker について提案する．5 節では，提案システムが合理的なプレイヤの元で正しく動作し，かつ，必ず終了することを示す．6 節では，プレイヤがデータ抽出のルールを入力するという選択肢がゲームにどのような影響を及ぼすかを検証する実験の説明をする．7 節は，まとめ

と今後の課題である．

2. 関連研究・システム

本研究には多くの関連研究とシステムがある．本節では，一般的な構造データの抽出に関する研究，GWAP を用いたデータ生成に関する研究，マイクロブログに関するまとめページやデータ抽出に関する研究について説明する．

2.1 構造データ抽出手法

1 節で説明したように，非構造データから構造データを抽出する方法は 3 つに大別される．(1) ラッパーなどの自動変換ソフトウェアを構築，(2) 自然言語処理技術により構造データを抽出，(3) 人の力によってデータを抽出．

(1) の例としては，Web データを XML に変換するラッパーを構築する研究 [1] がある．ラッパーを構築して抽出する方法は，抽出の際に人手が不要であるという利点がある．しかし，ラッパーを作成するためには十分な量のデータが必要であったり，構築そのものにコストがかかったりする事が問題である．

(2) の例としては，Wikipedia と Web の情報からオントロジーを自動構築する研究 [2] がある．自然言語処理により抽出する方法は，(1) と同様，抽出の際に人手が不要であるという利点がある．しかし，この方法はドメインによっては必ずしも正しい結果を得ることができない．例えば，Twitter 上の発言から筑波大学の学園祭の出店に関する構造データを抽出する場合，出店の店名や大学内の地区名のように辞書に掲載していないと思われる情報を抽出するのは難しい．

(3) の例としては，まとめサイトの管理者が抽出を行うように特定の人物が行うものと管理者が他の人々に依頼して抽出を行うものがある．これらは，人手で行うため幅広いドメインに対応できたり、情報の誤表記にも対応できたりと柔軟性が高い．また，前者は特定の人物で行われるためデータの品質を保ちやすい．しかし，前者はコストがかかり，大量のデータに対応するのは難しい．後者は，クラウドソーシングに代表されるように不特定多数の人物が行うものや比較的少数の協力者によって行われる手法まで含まれる．クラウドソーシングによるアプローチは多くの人々の参加によりデータを抽出するので各々の負担は小さいという利点がある．欠点としては，不特定多数の人物によりデータを抽出するため品質を保つことは比較的難しいといった欠点がある．

提案する Tweet Pecker は，(3) の手法の一つとしてとらえることができる．比較的少人数のグループから不特定多数のクラウドソーシングまで適用可能であるため，利用可能な人的リソースの規模に応じて柔軟に対応できるという利点がある．また，Tweet Pecker は同調ゲームによりデータを決定する，すなわち合意が得られたデータのみを採用するため，データの品質を保つことが期待できる．

2.2 GWAP を用いたデータ生成

GWAP によるデータ生成に関する代表的なシステムとして ESP Game がある．ESP Game で生成するデータは，画像に関するメタデータであり，構造データではない．それに対して本稿で提案する GWAP により生成されるデータは構造データである．GWAP により構造をもつデータを生成する研究には論文 [6] がある．論文 [6] では，GWAP によりオントロジーを構築する手法を提案している．本稿で提案する手法ではゲームのプレイヤはデータそのものを入力するだけでなく，データ抽出規則という “メタなデータ”を入力することが可能であるという点が異なる．また，本稿で提案する手法は後述するように構造データとして表データ (リレーション) を生成するという点が異なる．

2.3 マイクロブログに関する研究・システム

本研究のデータ抽出対象であるマイクロブログから情報を抽出する研究は数多く存在する [7] [8]．しかし，これらの研究は地域情報や関連語を抽出するといったように抽出するデータを限定していることが多く，柔軟性は低い．

また，マイクロブログの情報をまとめるシステムとして togetter [9] がある．togetter は，利用者によって指定された Twitter 上の発言を 1 本のリストに集めるシステムである．togetter でまとめられたデータは，発言がリストとして並べられているだけであり，発言の中からデータを抽出するわけではない．

3. 準備と問題

本節では，提案システムについて議論するために必要な準備を行い，我々が扱う情報抽出の問題を形式化する．まず，プログラミング言語 CyLog [10] の説明を行う．CyLog は，Datalog [11] に似た記法を持つ言語であり，人の入力を述語形式で容易に記述できる．次に，CyLog を用いて Tweet Pecker が扱う情報抽出の問題を形式化する．

4. CyLog

CyLog によるデータ処理のルールの例を次に示す．

```
Metadata(i, w)
  <- Image(i), Keyword(i, w)/open, inDict(w);
```

ここでは，`Image`，`Keyword`，`inDict` をそれぞれ画像，画像に対するキーワード，辞書を表す述語とする．また，これらの述語を用いて記述された `Image(i)` などの式を原子式 (Atom) と呼ぶ．原子式は左から評価され，引数に指定された変数が束縛される．ここで`/open` は未束縛変数の入力をプレイヤに問い合わせる指定であり，与えられた入力はあたかもリレーション中に存在していたかのように利用され，評価が行われる．これを特にオープン原子式 (Open Atom) と呼ぶ．したがって，このルールは各画像 `i` についてキーワードとなる語 `w` の入力を求め，その `w` が辞書に載っている時に，メタデータとして採用する事を表す．

5. 情報抽出の問題

CyLog を利用して，計算機もしくは人手を利用して Twitter 上の天気に関する発言をもつリレーション `Tweet(tw)` から構造データ `Weather(place, date, condition, time)` を抽出する問題を形式化できる．

まず，date, place, condition, time の値を抽出するアルゴリ

```
Weather(place, date, condition, time, tw)
   <- Tweet(tw), Extract_p(tw, place),
      Extract_d(tw, date),
      Extract_c(tw, condition),
      Extract_t(tw, time);
```

図 1　計算機による構造データの抽出

```
Weather(place, date, condition, time, tw)
   <- Tweet(tw), Extract_p(tw, place)/open,
     Extract_d(tw, date)/open,
     Extract_c(tw, condition)/open,
     Extract_t(tw, time)/open;
```

図 2　人手による構造データの抽出

ズムの存在を仮定すると，アルゴリズムによる抽出は図 1 のように形式化される．ここで，Tweet はツイートの集合を表す述語であり，Extract_x はそれぞれ date, place, condition, time の値を抽出するアルゴリズムに対応する述語である．

また，同様のデータを人手により抽出する場合は，図 2 のように形式化される．

Tweet Pecker は，GWAP を用いて図 2 のプログラムを徐々に図 1 のプログラムに変換する仕組みである．

6. 提案システム Tweet Pecker

本節では，GWAP による構造データ抽出システム Tweet Pecker の説明を行う．

6.1 TweetPecker の概要

Tweet Pecker は，GWAP により Twitter 上の発言（以下**ツイート**と表記）からリレーションを抽出するシステムである．ここで，ツイートは発言内容だけでなく発言者，発言日時などの情報をもつ．以降では，GWAP を行う人々(2 人以上) の各自を**プレイヤ**と呼ぶ．Tweet Pecker の特徴は，後述するように，最初はプレイヤによる情報抽出 (オープン原子式による抽出) のみから始まるが，プレイヤの動きによって徐々に計算機によって抽出 (オープンでない原子式による抽出) が行われていくよう，GWAP がデザインされていることである．

TweetPecker による構造データ抽出の流れを図 3 に示す．以下はその説明である．

（1） システム利用開始時に必要な入力は，対象となるツイート集合を表すハッシュタグと，抽出するリレーションのスキーマ $St(a_1, a_2, \ldots, a_n, \texttt{tw})$(例えば，Weather(place, date, condition, time, tw)) である (図 3(1))．ここで，tw はツイートを格納する属性である．

（2） 入力に基づき，そのハッシュタグを含むツイートの集合を表すリレーション Tweet(tw) をまず作成し，そこから指定されたスキーマのリレーションを作成する (図 3(2))．例えば，システム利用者がハッシュタグ #天気とリレーションのスキーマ Weather(place, date, condition, time, tw)(tw は省略可) を入力すると，“#天気”というハッシュタグを含むツイート集合を表すリレーション Tweet(tw) を作成し，リレーション Weather を作成する．

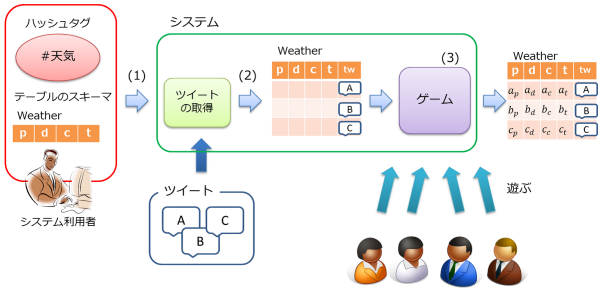

図 3　Tweet Pecker による構造データ抽出の流れ

（3） その後，プレイヤにゲームを行わせることでリレーション Weather の各属性値を決定する (図 3(3))．

プレイヤによる入力は次の 2 通りである．(a) 属性値 v の入力，(b) 計算機が属性値を抽出するためのルール r(以下**抽出規則**と表記)．抽出規則の入力を許すことにより，最初は，各属性値の抽出は図 1 のようにオープン原子式 Extract_a_i(tw, a_i)/open により人手で行われるが，抽出規則の増加とともに，その抽出は図 2 のようにオープンでない原子式 Extract_a_i(tw, a_i) によって実行されるようになる．抽出規則の詳細は 6.4 節で説明する．

6.2 構造データ抽出ゲームの概要

Tweet Pecker における構造データ抽出ゲームの概要を説明する．以下の説明では，5 節で説明した Weather(place, date, condition, time, tw) の例を用いて説明する．

Tweet Pecker による構造データ抽出は同調ゲームによって行われる．同調ゲームとは，プレイヤの振る舞いが同じであるときにポイントを与えるという構造を持つゲームである．このゲームは，Tweet(tw) から，スキーマ $St(a_1, a_2, \ldots, a_n, \texttt{tw})$ (すなわち Weather(place, date, condition, time, tw)) で表現される構造データを抽出する．

Tweet Pecker ではプレイヤ p_1, p_2 が Tweet(tw) のタプル t_k に束縛されるいずれかのオープン原子式 (例えば Extract_p(tw, place)/open) に対して，両者ともに同じ属性値 v を入力すると，p_1, p_2 両者にポイントが与えられ，同時に Weather(place, date, condition, time, tw) の属性値として v を格納する．

また，Tweet Pecker では，プレイヤは属性値 v の入力だけでなく，オープンでない原子式 (Extract_p(tw, place) 等) を実現するための抽出規則 r を入力することが可能である．抽出規則 r が入力されると，その規則が適用できる場合にはオープンでない原子式が，構造データ抽出に利用されるようになる．適用できる規則が存在しない場合には，常にオープン原子式 (Extract_p(tw, place)/open) だけが利用される．抽出規則の入力を許すことにより，属性値の抽出が徐々に計算機によって行われるようになる．

先述の通り，Tweet Pecker ではプレイヤは属性値の入力だけでなく，抽出規則も入力することが可能である．しかし，説明のため，まずは，抽出規則 r を入力することを許さない場合のゲーム，すなわちプレイヤの選択肢を属性値の入力に限定したゲーム (以下**ゲーム A**) について説明する．その後，属性値

の入力に加え，抽出規則の入力を許可するゲーム (以下**ゲーム A'**) について説明する．

6.3 ゲーム A: プレイヤの選択肢を属性値の入力に限定

ゲーム A は，図 2 のオープン原子式を同調ゲームによって評価することに該当する．

プレイヤの集合を $P = \{p_1, p_2, \ldots, p_n\}$ とし，ツイートの集合をリレーション `Tweet(tw)` とする．抽出結果を格納するリレーションは `Weather(place, date, condition, time, tw)` とする．

この時，**ゲーム A** の流れは以下のようになる．

（1） システムが，`Tweet(tw)` のタプル t_k をプレイヤ p_i に割り当てる．ここで，どのようにタプル t_k とプレイヤ p_i を決定するかは 6.3.1 節で説明する．

（2） プレイヤ p_i は，タプル t_k に束縛されるそれぞれのオープン原子式 (`Extract_p(tw, place)` 等) に対して属性値 v_p, v_d, v_c, v_t を入力する．v_x はそれぞれ `place, date, condition, time` に対応する属性値である．

（3） プレイヤ p_i が属性値の入力を終えるとタプル t_k の割り当てを解除する．

（4） 同じタプル t_k に対して，他のプレイヤ p_j $(j \neq i)$ が入力した値 v'_x が (2) で入力した v_x と等しい場合 $(v_x = v'_x)$，タプル t_k に対する x の属性値を v_p に決定する．

（5） 格納された属性値 v_x, v'_x を入力したプレイヤ p_i, p_j にポイントを与える．

（6） `Weather(place, date, condition, t)` の属性値が全て決定するまで (1) から (5) を繰り返す．

続いて，(1) と (2) の詳細をそれぞれ説明する．(5) の詳細は，6.5 節で説明する．

6.3.1 プレイヤへのタプル割り当ての詳細

各プレイヤに `Tweet(tw)` のタプルを割り当てる方法を説明する．システムはタプルが 1 つも割り当てられていないプレイヤ p_i に対してタプルを 5 つ割り当てる．ここで，タプルが 1 つも割り当てられていないプレイヤとは次のいずれかの状況である．(1) ゲームへ参加したばかりのプレイヤ．(2) 割り当てられたタプルの属性値を入力し終え，全てのタプルの割り当てが解除されたプレイヤ．

プレイヤ p_i に割り当てるタプルは以下の条件を満たしたタプルである．

条件 1 そのプレイヤ p_i が属性値を入力したことがない．

条件 2 少なくとも 1 つは属性値が決定していない．

条件 3 その時点で，他のプレイヤ $\{p_j | p_j \in P, p_i \neq p_j\}$ に割り当てられていない．

条件を満たすタプルが存在しない場合は，そのプレイヤ p_i にタプルは割り当てられない．また，プレイヤ p_i には他のプレイヤ $\{p_j | p_j \in P, p_i \neq p_j\}$ によって属性値が入力されたことのあるタプルが優先的に割り当てられる．これは，タプルの数がプレイヤの人数に対して非常に多い場合，属性値が入力されたことのあるタプルがプレイヤになかなか割り当てられない事態を避けるためである．

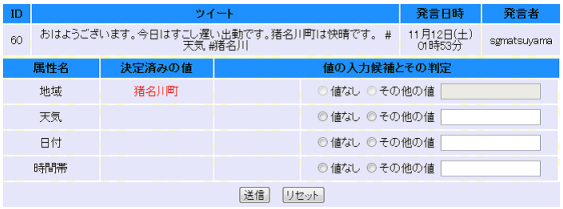

図 4 ゲーム A における属性値入力フォーム

6.3.2 属性値の入力

Tweet Pecker では，属性値の入力に Web フォームを用いる (図 4)．プレイヤには，割り当てられたタプル 5 つのうち 1 つがランダムに表示される．プレイヤは表示されたタプルの各属性値を図 4 の選択肢から選び入力する．抽出する値が存在する場合は，「その他の値」を選択し，その横のテキスト欄に属性値を自由記述で入力する．以降，属性値を図 4 のテキスト欄に自由記述で入力することを**直接入力する**という．抽出する値が存在しない場合は，「値なし」を選択する．ただし，既に属性値が決定しているものは入力できない (例えば図 4 の地域属性)．

6.4 ゲーム A': プレイヤの選択肢として抽出規則の入力を許可

Tweet Pecker における抽出規則とは計算機が属性値を抽出するためのルールである．例えば次のようなものがある．

(**規則例 1**) ツイートの発言内容に/夕立/ならば，時間帯属性の値は夕方である．

Tweet Pecker で扱う抽出規則は次の 4 つの要素から構成される．

要素 1: マッチング条件 ツイート中にどのような文字列が含まれるかを表す要素．これは，正規表現を用いて記述する．規則例 1 の場合 "/夕立/" となる．

要素 2: 対象テキスト マッチング条件をツイートのどの情報に適用するかを決定する要素．発言内容，発言者名，発言日時の 3 つから選択可能とする．規則例 1 の場合，"発言内容" である．また，**対象テキスト**を発言内容にする場合，**マッチング条件**をテキスト全体に適用するか，テキスト中の各名詞に適用するかを選択できる．各名詞にマッチング条件を適用することを許すことにより，例えば "/.*県/" というマッチング条件にした場合，ツイートのテキスト中から「茨城県」という名詞にマッチさせることが可能となる．

要素 3: 抽出値属性 属性値として抽出する値の属性名である．規則例 1 の場合，"時間帯" である．

要素 4: 抽出値 属性値として抽出する値である．規則例 1 の場合，"夕方" になる．この値は**マッチング条件**によってマッチした箇所，またはその一部にすることが可能である．例えば，「ツイートのテキストを単語に分割した際，単語の 1 つが/.*県/ならば，地域属性の値はマッチした箇所である」という抽出規則を作成すると「茨城県」や「千葉県」などの県名を地域の属性値として抽出することが可能である．

抽出規則の作成は属性値の入力と同様，Web フォームを用いる (図 5)．抽出規則の作成による属性値の抽出は次のように行われる．まず，**マッチング条件**と**対象テキスト**から，`Tweet(tw)`

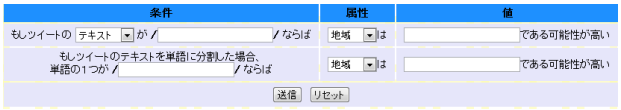

図 5　抽出規則入力フォーム

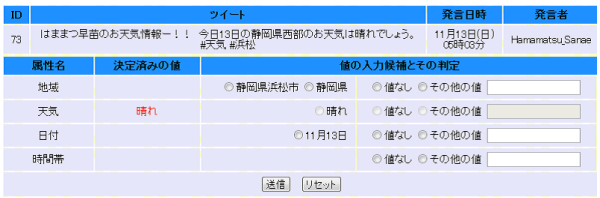

図 6　ゲーム B における属性値入力フォーム

のタプル t_k を決定する．このタプルは規則によっては複数存在する．そして**抽出値属性**から，抽出値がタプル t_k に束縛されるどの原子式 (`Extract_p(tw, place)`, `Extract_d(tw, date)`, `Extract_c(tw, condition)`, `Extract_t(tw, time)`) に対応するかを決め，**抽出値**を属性値として抽出する．

6.4.1　属性値の入力

抽出規則により計算機が値が抽出するとプレイヤによる属性値入力の際，その値がフォームに表示される (図 6)．プレイヤは属性値を入力する際，この値を選択することが可能になる．つまり，3 節の用語で説明すると，ゲーム B ではオープン原子式とオープンでない原子式の両方を同調ゲームによって評価する．ただし，自分が作成した抽出規則により抽出された値が入力の選択肢に含まれる属性は，入力することができない．

6.5　ポイントの付与

システムは，各属性値が決定したとき，プレイヤにポイントを与える．与えるポイントは次の通りである．

- 決定した属性値が「値なし」の場合，その値を入力したプレイヤ p_i と p_j に 3 点与える．
- 決定した属性値が直接入力による値の場合，その値を入力したプレイヤに 5 点与える．
- 決定した属性値が抽出規則により抽出された値の場合，その値を入力したプレイヤ p_i と p_j に 3 点与え，その抽出規則を最初に作成したプレイヤ p_k に 2 点与える．例えば，「おはようございます。つくば市の天気は晴れ」というツイートのタプル t が存在するとする．この時，p_1 が「もしツイートの発言内容が/おはよう/なら時間帯は朝である」という抽出規則 r_1 を入力し，その後，p_2 が抽出規則 r_2「もしツイートの発言内容が/おはようございます/ならば時間帯は朝である」を入力したとする．r_1, r_2 により抽出された値「朝」が p_3, p_4 の入力によって属性値として決定したとき，ポイントは p_1, p_3, p_4 に与えられる．
- 決定した属性値以外の値を抽出する抽出規則を作成したプレイヤを 1 点減点する．

Tweet Pecker では，決定した属性値以外の値を抽出する抽出規則を作成したプレイヤを 1 点減点することにより，誤った属性値が多く抽出されることを避けている．これは誤った属性値を多く抽出する規則が増えるとプレイヤが属性値を入力する際，表示される属性値が増え，プレイヤが値を入力する妨げになるからである．

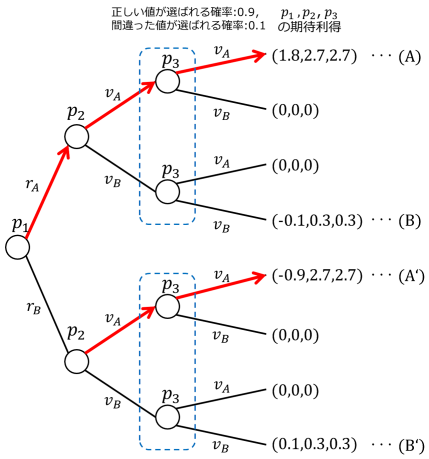

図 7　Tweet Pecker のゲーム木

7.　ゲームの妥当性

本節では，本研究で提案するゲームの妥当性を議論するため，合理的なプレイヤの元でゲームが次を満たすことを示す．(1) 正しいデータとルールが入力される．(2) 必ずゲームが終了する．

7.1　入力された値と抽出規則の正しさ

[**定理** 1]　抽出規則を入力するプレイヤを p_1, 属性値を入力するプレイヤを p_2, p_3 とする．この時，ゲームの解は正しい抽出規則と属性値を入力することである．□

証明. 図 7 は Tweet Pecker による 3 人のゲームを図示したゲーム木 [12] である．ここで，終点は各プレイヤの期待利得を表し，p_3 に関する情報集合 (点線) は，p_3 が p_2 の手を知らないことを示す．v_A は，正しい属性値であり，v_B は，誤った属性値である．また，r_A は，v_A を抽出する抽出規則であり，r_B は v_B を抽出する抽出規則である．プレイヤ p_1 は r_A，もしくは r_B を入力するとし，p_2 と p_3 はそれぞれ v_A か v_B のどちらかを入力するとする．実際には，誤った属性値は複数存在するが，冗長であるため，ここでは誤った属性値は v_B のみを扱う．

p_2 と p_3 が行う部分ゲームは，6.2 節で説明した同調ゲームであり，同調した場合に得られる利得はどの選択肢でも同じである (図 7 の (A) と (B)，(A') と (B'))．しかし，選択肢のうち，正解になりやすいものは選ばれる可能性が高いので，利得の期待値は異なる．したがって，p_2 と p_3 が行う部分ゲームの解が決まり (図 7 の (A)，(A'))，さらに，p_1 は彼らが選ぶ属性値を出力する抽出規則を入力することになる (図 7 の (A))．□

7.2　ゲームの終了性

[**定理** 2]　対象となるツイート集合が有限の時，TweetPecker は必ず終了する．□

証明. ツイート集合のサイズを N とし，抽出する構造データの属性 (ツイートを格納する属性は除く) の数を M とする．Tweet Pecker は一般的な同調ゲームと異なり，プレイヤは属性値の入力と抽出規則の入力と 2 つの行動が可能である．しかし，ゲー

ムのデザインにより抽出規則の入力だけでは属性値は決定しないため，ゲームの終了には少なくとも $2MN$ 回の属性値の入力が必要になる．しかし，抽出規則は一般に無限にあるため，全てのプレイヤが永遠に抽出規則を入力し続ける可能性がある．その場合，ゲームは終了しない．

TweetPecker では，6.5 節で説明したように，ある抽出規則によって計算機が抽出した値が属性値に決定した際，その抽出規則を作成したことによるポイントが与えられるプレイヤはただ 1 人である．つまり，ある 1 つの属性値が決定したとき,「利得が得られる」抽出規則もただ 1 つである．したがって，対象となるツイート集合が有限の時,「利得が得られる」抽出規則の集合も必ず有限であり，合理的なプレイヤはいつかは必ずルールではなく属性値を入力する．したがって，対象となるツイート集合が有限の時，Tweet Pecker のゲームは必ず終了する．

8. 実　　験

本節では，本研究で行った 2 つの実験について説明する．

8.1 実験 1:抽出規則の入力を許すことによる影響の調査

実験 1 では，抽出規則という “メタなデータ”を入力することが，GWAP による構造データ抽出に対してどのような影響を及ぼすかを調査した．

8.1.1 実 験 方 法

6.3 節で説明したゲーム A と 6.4 節で説明したゲーム B の 2 つを用意する．各ゲームにつき，プレイヤを 6 人用意してゲームを行わせる．ツイート集合と抽出する構造データのスキーマは両ゲームとも同じとする．ゲームは 60 分間行い，プレイヤの振る舞いをログデータに記録する．実験終了後に，ログデータを分析し，各ゲームの決定した属性値の数やプレイヤの得点分布を比較する．

8.1.2 実験データ

今回の実験では，入力であるリレーションのスキーマを Tenki(地域，天気，日付，時間帯)，ハッシュタグを “#天気” とする．今回，このようなリレーションのスキーマとハッシュタグにしたのは Tweet Pecker の利用に慣れていない被験者が抽出規則を比較的作成しやすいテーマであると考えられるからである．実験に使用するツイートは 670 件用意した．

8.1.3 実験結果と考察

実験結果を表 1 および図 8〜図 12 に示す．

決定した属性値の数．各ゲームにおいて決定した属性値の数について説明する．表 1 は，実験によって決定した属性値の数等をゲーム毎に比較した表である．ここで，属性値に決定した入力の割合とはプレイヤが入力した属性値の数全体に対してプレイヤが入力した属性値がテーブルの属性値に決定した値の数の割合のことである．図 8 は決定した属性値の数の累積である．縦軸が決定した属性値の数であり，横軸は経過時間である．表 1 から分かる通り，ゲーム B の方がより多くの属性値が決定された．また，図 8 に示してあるように，決定した属性値の数はゲーム序盤ではゲーム A の方が多いが，時間が経過するとともにゲーム B の方が多くなっていることが分かる．ゲーム B で序盤に決定した属性値が少ないのは，抽出規則の作成により属性値の入力のペースが遅く，したがって属性値の決定のペースも遅いからと考えられる．その後，抽出規則の作成とともに，プレイヤは直接入力ではなく計算機が抽出した値を選択することが多くなるので，入力のペースが速くなる．これは，選択肢から選んで入力する方が直接入力するよりも速く入力できるからである．したがって徐々に属性値の決定のペースが速くなったと考えられる．

表 1　決定した属性値の数やプレイヤの属性値の入力に関するデータ

項目 \ ゲーム名		ゲーム A	ゲーム B
決定した属性値の数		904	1054
	直接入力による値の数	639	126
	計算機が抽出した値の数	0	818
	属性値が「値なし」である数	265	110
プレイヤが入力した属性値の数		2751	2707
属性値に決定した入力の割合		66%	78%

また，ゲーム B ではゲーム A に比べ属性値の入力の数全体に対して決定した属性値の数の割合が高い．これはゲーム A では直接入力による属性値の入力が多く，直接入力した値は他のプレイヤに表示されないため，入力した値が他のプレイヤと同じ内容であっても異なる値を入力する可能性があるからと考えられる．例えばプレイヤ p_i がゲーム A においてある属性値に対して「晴れ」と入力し，他のプレイヤ p_j がその属性値に対して「晴」と入力した場合は決定しない．しかし，ゲーム B では値を選択することが多いので「晴れ」という選択肢がある場合，プレイヤ p_i と p_j の両者はどちらも「晴れ」を選択すると思われる．したがってゲーム B ではゲーム A に比べ全体の入力数に対して決定する属性値の数の割合がが多いと考えられる．

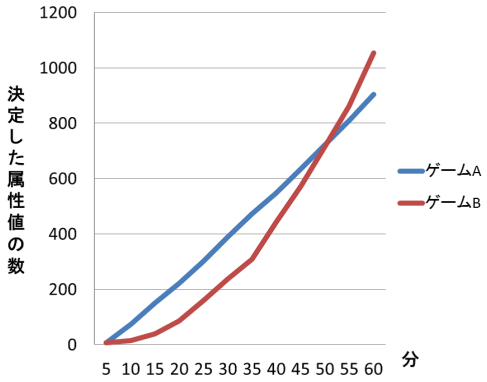

図 8　決定した属性値の数

直接入力と計算機による抽出の割合の変化．次に，ゲーム B において人が直接入力した値と計算機が抽出した値の割合について説明する．図 9 はゲーム B において決定した属性値が，人が直接入力した値なのか計算機が抽出した値なのかの内訳を表している．ただし，決定した属性値の内,「値なし」であるものは除外している．図 9 からゲーム開始直後は直接入力された値，すなわち人手によって抽出された値が多いが，徐々に計算機によって抽出された値が多くなっていることがわかる．また，時間の経過とともに計算機によって抽出された値が多くなるもの

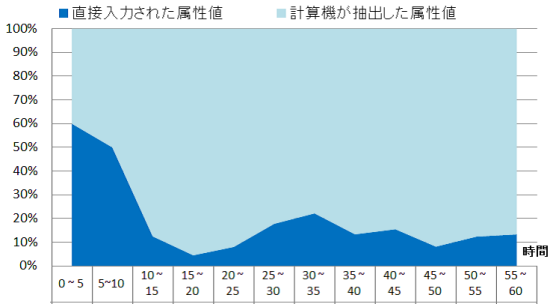

図 9　ゲーム B における決定した属性値の内訳

の，人手による抽出された値も 10%から 20%ほどあることから人手による抽出も必要であることがわかる．

ポイントの分布．次に，各ゲームにおいてプレイヤが得たポイントについて説明する．図 10 はゲーム A における時間毎の各プレイヤのポイントの累積を表している．横軸は経過時間である．同様に，図 11 はゲーム B における時間毎の各プレイヤのポイントの累積を表している．図 12 は各プレイヤに付与されたポイントの内訳である．図 12 のグラフ中の各項目を説明する．

（1） 直接入力による得点．プレイヤがある属性値に対して直接入力し，その値が決定した場合に付与されるポイント．

（2） 選択肢の値を入力することによる得点．プレイヤが選択肢の値を選んで入力し，その値が決定した場合に付与されるポイント．選択肢の値とは,「値なし」もしくは計算機が抽出した値である．

（3） 作成した抽出規則による加点．プレイヤが抽出規則を作成し，その規則によって抽出された値が属性値として決定した場合に付与されるポイント．

（4） 作成した抽出規則による減点．プレイヤが抽出規則を作成し，その規則によって抽出された値が誤っていた場合に引かれるポイント．

図 10 と図 11 からわかるように，ゲーム B ではゲーム A に比べ，各プレイヤのポイントに差が見られる．これは，ゲーム A ではプレイヤは 1 つの属性値に対して直接入力もしくは「値なし」という 2 つの選択肢しかないため，各プレイヤのポイントに大きな差が見られなかったからと考えられる．図 12 からゲーム A では各プレイヤが得たポイントの内訳も似たような割合であることがわかる．一方，ゲーム B ではポイントを得る手段に抽出規則を作成するという方法が加わるため，プレイヤによって行動が異なった．これは，各プレイヤのポイントの内訳がプレイヤによって違いが見られることからもわかる．したがって各プレイヤの得点に差が見られたと考えられる．

先述の通り，ゲーム B では決定した属性値の多くは計算機によって抽出された値である．それにもかかわらず，図 12 からわかるように全体として抽出規則の作成による加点の割合が小さい．ゲームとしては正しい属性値を抽出する抽出規則ができるだけ多く入力されることが望ましい．したがって，プレイヤに付与するポイントの大きさを改善する必要があると思われる．

また，プレイヤ ID が 10 のプレイヤは，選択肢の値を入力することによる得点が大きく，ゲーム B において最もポイントが

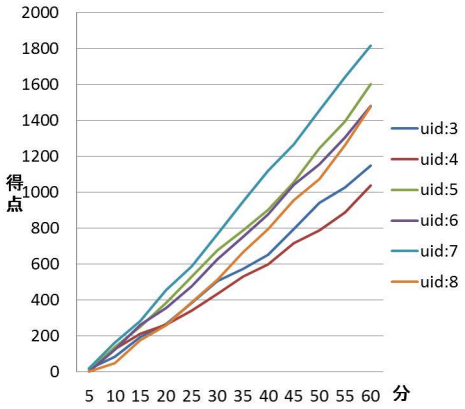

図 10　ゲーム A のプレイヤの得点分布

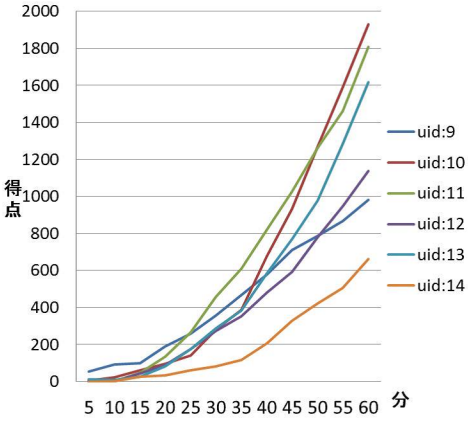

図 11　ゲーム B のプレイヤの得点分布

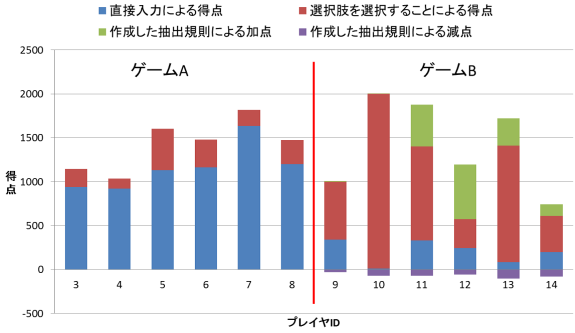

図 12　各プレイヤのポイントの内訳

高いプレイヤである．実験終了後にプレイヤ ID が 10 のプレイヤにゲーム中の振る舞いの方針を聞いてみると，表示されたツイートは見ずに何等かの選択肢をランダムに入力していたことがわかった．このような振る舞いをしたプレイヤが最もポイントが高くなるのは好ましくない．したがって，ある決定した属性値に対して別の値を入力したプレイヤは減点するといった対策が必要である．

8.2　実験 2:抽出したデータの正しさ

次に，本稿で提案するゲームで抽出したデータが正しいデータであるかの調査を行った．

8.2.1　実験方法・実験データ

実験 2 では，実験 1 で行ったゲーム B がツイートから正しく

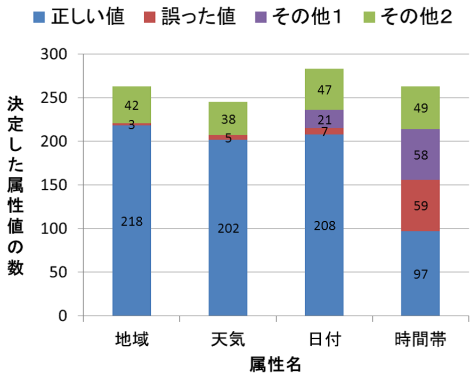

図 13　抽出した属性値の正しさ

抽出できているかの調査を著者自身が行った．決定した属性値を正しい値と誤った値，正誤にわけられないもの (原因によってその他 1 とその他 2 の 2 つに分類) の 4 つに分類した．その他 1 と 2 はそれぞれ次のようなものである．(その他 1) 今回の実験では，プレイヤに表示するツイートの投稿日時が日本時間ではなかったため，ツイートの内容と投稿日時に矛盾が生じてしまったツイートが存在した．このようなツイートから抽出した属性値は正しく抽出した値かどうかの判断が難しいため，それらはその他 1 とした．(その他 2) 1 つのツイートに対して複数のタプルが生成されるべきツイートが存在した．例えば,「東京の天気は雨，大阪では晴れ」というツイートがあった場合，2 つのタプルが存在すべきである．しかし，Tweet Pecker では 1 つのツイートにつき 1 タプルしか生成されない．このようなツイートに対して決定した属性値が正しい値かどうかの判断は難しいため，それらはその他 2 とした．

8.2.2　実験結果と考察

図 13 は今回の実験で抽出したリレーションの各属性ごとに決定した属性値の正しさを表している．図 13 からわかるように 4 つの属性の内，時間帯以外は誤った値を抽出することはほとんどなく，概ね正しい値を抽出することができた．また，いずれの属性も 15%から 20%ほどその他 2 であることから 1 つのツイートから複数のタプルを生成することができる仕組みが必要であることがわかる．時間帯属性に誤りが多かった理由は次の 2 つが考えられる．今回の実験で使用したハッシュタグ “#天気” によって取得したツイートには大きく分けて 2 種類ある．1 つは投稿時の天気に関するツイートであり，もう 1 つは天気予報のツイートである．誤った値の多くは天気予報のツイートに対して時間帯の属性値が投稿時の時間帯に決定していた．これは天気予報のツイートでは時間帯について述べているツイートは少なかったため，時間帯の値として発言時の時間帯を入力する傾向があったと考えられる．また，実験前に被験者に対して各属性について十分な説明をしていなかったため，時間帯を天気の時間帯ではなくツイートを投稿した時間帯と誤った認識でゲームを行っていた可能性も考えられる．

9.　まとめと今後の課題

本稿では，非構造データから構造データを抽出する方法として GWAP を用いる手法を提案した．提案手法を実現したシステム Tweet Pecker は，Twitter 上の発言からリレーションを抽出するシステムである．Tweet Pecker はシステム利用者にリレーションのスキーマを入力して受け取り，抽出するリレーションの属性値を複数プレイヤによる同調ゲームによって決定する．Tweet Pecker の特徴は，プレイヤに対して属性値の入力をさせるだけでなく，抽出規則の入力を許すことである．このため，最初は人手によって属性値が抽出されるが，抽出規則が増えると属性値の抽出が徐々に計算機によって行われるようになる．本稿ではこのゲームが合理的なプレイヤの元で正しく動作し，かつ必ず終了することを示した．また，実験により，プレイヤが抽出規則を入力可能なゲームの振る舞いを検証し，ゲームの進行につれて計算機による抽出の割合が増加する事を確認した．

今後の課題の 1 つは，8.2 節で説明したように，Tweet Pecker で抽出する構造データにおいて 1 つのツイートにつき複数のタプルを生成することを可能にするなど，提案手法が扱える対象をより拡大する事である．

謝　　辞

本研究の一部は JST さきがけ「情報環境と人」および科学研究費補助金 (#21240005) による．

文　　献

[1] Wei Han, David Buttler, Calton Pu. “Wrapping web data into XML”. Newsletter ACM SIGMOD Record. 2001, 30(3), pp.33-38.
[2] 白川 真澄, 中山 浩太郎, 荒牧 英治, 原 隆浩, 西尾 章治郎. “Wikipedia と Web の情報を組み合わせたオントロジー構築の試み”. 電子情報通信学会論文誌. D, 情報・システム. 2011, J94-D(3), pp.525-539.
[3] Luis von Ahn. “Invited Talk: Human computation”. International Conference On Knowl-edge Capture. Proceedings of the 4th international conference on Knowledge capture. 2007, pp.5-6.
[4] Luis von Ahn, Laura Dabbish. “Designing Games With A Purpose”. Communication of the ACM. 2008, 51(8), pp.58-67.
[5] Luis von Ahn , Laura Dabbish, “Labeling Images with a Computer Game”. Proceedings of the SIGCHI conference on Human factors in computing systems. 2004, pp.319-326.
[6] 安永ゆい, 望月祥司, 森嶋厚行. “GWAP によるオントロジ構築手法の提案”. 全国大会講演論文集. 2011, 2011(1), pp.765-767.
[7] 中本聖也, 北野光一, 寺口敏生, 田中成典, 西江将男. “マイクロブログからの地域の話題抽出に関する研究”. 全国大会講演論文集. 2011, 2011(1), pp.783-785.
[8] 岩井一晃, 鈴木優, 石川佳治. “マイクロブログにおける関連語の自動抽出”. 全国大会講演論文集. 2011, 2011(1), pp.679-681.
[9] “togetter”. http://togetter.com/.
[10] Atsuyuki Morishima, Norihide Shinagawa, Shoji Mochizuki. “The Power of Integrated Abstraction for Data-centric Human/Machine Computations”. VLDS 2011. 2011, pp.5-9.
[11] Hector Garcia-Molina, Jeffrey D Ullman, Jennifer Wisdom. “Database Systems: the Complete Book”. Prentice Hall. 2002, pp.463-502.
[12] F. Vega-Redondo. Economics and Theory of Games. Cambridge University Press, 2003.